/routes/users.js

const express = require('express');

const router = express.Router();

const User = require('../models/user');

const catchAsync = require('../utils/catchAsync');

const passport = require('passport');

router.get('/register', (req, res) => {

    res.render('users/register');

})

router.post('/register', catchAsync(async (req, res, next) => {

    const { user } = req.body;

*try* {

        const newUser = new User({ username: user.username, email: user.email });

        const registerUser = *await* User.register(newUser, user.password)

        req.login(registerUser, err => {

*if* (err) {

*return* next(err);

            }

        })

        req.flash('success', 'Welcome to Yelp-Camp');

        res.redirect('/campgrounds');

    }

*catch* (e) {

        req.flash('error', e.message);

        res.redirect('/register');

    }

}))

router.get('/login', (req, res) => {

    res.render('users/login');

})

router.post('/login', passport.authenticate('local', { failureFlash: true, failureRedirect: '/login' }), (req, res) => {

    req.flash('success', 'Welcome back !!!');

    res.redirect('/campgrounds');

})

router.get('/logout', (req, res) => {

    req.logOut();

    req.flash('success', 'Goodbye');

    res.redirect('/');

})

module.exports = router;

middleware

const res = require("express/lib/response")

module.exports.reqLogin = (req, res, next) => {

*if* (!req.authenticate) {

        req.flash('error', 'You must login to access')

        res.redirect('/campgrounds');

    }

*else* {

        next();

    }

}

App.js

const express = require('express');

const app = express();

const path = require('path');

const mongoose = require('mongoose');

const methodOverride = require('method-override');

const ejsMate = require('ejs-mate');

const { nextTick } = require('process');

const { runInNewContext } = require('vm');

const ExpressError = require('./utils/ExpressError');

const { STATUS\_CODES } = require('http');

const campgroundRoutes = require('./routes/campgrounds');

const reviewRoutes = require('./routes/reviews');

const userRoutes = require('./routes/users')

const session = require('express-session');

const flash = require('connect-flash');

const localStrategy = require('passport-local');

const passport = require('passport');

const User = require('./models/user');

mongoose.connect('mongodb://localhost:27017/yelp-camp1', (() => {

    console.log("connect to DB")

}), (e => {

    console.error(e.message);

}));

app.set('view engine', 'ejs');

app.set('views', path.join(\_\_dirname, 'views'));

app.use(express.static(path.join(\_\_dirname, 'public')));

app.use(express.urlencoded({ extended: true }));

app.use(methodOverride('\_method'));

app.engine('ejs', ejsMate);

const sessionconfig = {

    secret: 'thisisasecret',

    resave: false,

    saveUninitialized: true,

    cookie: {

        expires: Date.now() + 1000 \* 60 \* 60 \* 24 \* 7,

        httpOnly: true,

        maxAge: 1000 \* 60 \* 60 \* 24 \* 7,

    }

}

app.use(session(sessionconfig));

app.use(flash());

app.use(passport.initialize());

app.use(passport.session());

passport.use(new localStrategy(User.authenticate()));

passport.serializeUser(User.serializeUser());

passport.deserializeUser(User.deserializeUser());

app.use((req, res, next) => { *//used before other middleware*

    res.locals.currentUser = req.user;

    res.locals.success = req.flash('success');

    res.locals.error = req.flash('error');

    next();

})

app.use('/campgrounds', campgroundRoutes);

app.use('/campgrounds/:id/reviews', reviewRoutes);

app.use('/', userRoutes);

app.get('/host', async (req, res) => {

    const host = new User({ username: 'OSG', email: 'osg@yelp-camp.com' });

    const registerHost = *await* User.register(host, 'osg');

    res.send(registerHost);

})

app.get('/', (req, res) => {

    res.render('home')

})

*// app.get('/makeCampground', (req, res) => {*

*//     const newCampground = new Campground({*

*//         title: "My back yield",*

*//         description: "Cheap & available",*

*//     });*

*//     res.send(newCampground);*

*// })*

app.all('\*', (req, res, next) => {

    next(new ExpressError(404, 'Page not found'))

})

app.use((err, req, res, next) => {

*// const { statusCode = 500, message = 'Something went wrong' } = err;*

*// res.status(statusCode).send(message);*

    const { statusCode = 500 } = err;

*if* (!err.message) { err.message = 'Unspecific Errors' }

    res.status(statusCode).render('error', { err })

})

app.listen(3000, () => {

    console.log("Listen to port 3000")

})

model/campground.js

const mongoose = require('mongoose');

const review = require('./review');

const Schema = mongoose.Schema;

campgroundschema = new Schema({

    title: String,

    price: Number,

    location: String,

    image: String,

    description: String,

    reviews: [{

        type: Schema.Types.ObjectId,

        ref: 'Review'

    }]

})

campgroundschema.post('findOneAndDelete', async function (doc) {

*await* review.deleteMany({

        \_id: {

            $in: doc.reviews

        }

    })

})

*// campgroundschema.post.findOneAndDelete,async(function(x){*

*//     review.remove(*

*//         id: {$in: x.\_id}*

*//     )*

*// })*

const Campground = mongoose.model('Campground', campgroundschema);

module.exports = Campground;

views/users/register.js

<% layout('layouts/boilerplate') %>

    <h2 *class*="text-center">Yelp-Camp Registration</h2>

    <form *action*="/register" *method*="POST" *class*="validate-form" *novalidate*>

        <div *class*="mb-3 row">

            <label *for*="username" *class*="col-sm-2 col-form-label">User Name</label>

            <div *class*="col-sm-10">

                <input *type*="text" *class*="form-control" *id*="username" *name*="user[username]" *placeholder*="Username"

*required*>

            </div>

        </div>

        <div *class*="mb-3 row">

            <label *for*="email" *class*="col-sm-2 col-form-label">Email</label>

            <div *class*="col-sm-10">

                <input *type*="email" *class*="form-control" *id*="email" *name*="user[email]"

*placeholder*="username@yelp-camp.com" *required*>

            </div>

        </div>

        <div *class*="mb-3 row">

            <label *for*="inputPassword" *class*="col-sm-2 col-form-label">Password</label>

            <div *class*="col-sm-10">

                <input *type*="password" *class*="form-control" *id*="inputPassword" *placeholder*="Password"

*name*="user[password]" *required*>

            </div>

        </div>

        <div *class*="d-grid gap-2 col-2 mx-auto">

            <button *class*="btn btn-primary">Sign-Up</button>

        </div>

    </form>

views/users/login.js

<% layout('layouts/boilerplate') %>

    <h2 *class*="text-center">Yelp-Camp Login</h2>

    <form *action*="/login" *method*="POST" *class*="validate-form" *novalidate*>

        <div *class*="mb-3 row">

            <label *for*="username" *class*="col-sm-2 col-form-label">User Name</label>

            <div *class*="col-sm-10">

                <input *type*="text" *class*="form-control" *id*="username" *name*="username" *placeholder*="Username" *required*>

            </div>

        </div>

        <div *class*="mb-3 row">

            <label *for*="inputPassword" *class*="col-sm-2 col-form-label">Password</label>

            <div *class*="col-sm-10">

                <input *type*="password" *class*="form-control" *id*="inputPassword" *placeholder*="Password" *name*="password"

*required*>

            </div>

        </div>

        <div *class*="d-grid gap-2 col-2 mx-auto">

            <button *class*="btn btn-success">Login</button>

        </div>

    </form>

Routes/campground

const express = require('express');

const router = express.Router();

const catchAsync = require('../utils/catchAsync');

const ExpressError = require('../utils/ExpressError');

const Campground = require('../models/campground');

const flash = require('connect-flash');

const { campgroundSchema } = require('../schema.js');

const { reqLogin } = require('../middleware');

const validateCampground = (req, res, next) => {

    const { error } = campgroundSchema.validate(req.body);

*if* (error) {

        const msg = error.details.map(el => el.message).join(',')

*throw* new ExpressError(400, msg)

    }

*else* {

        next();

    }

}

router.get('/', async (req, res) => {

    const campgrounds = *await* Campground.find();

    res.render('campgrounds/index', { campgrounds })

})

router.get('/new', (req, res) => {

    res.render('campgrounds/new');

})

router.get('/:id/edit', reqLogin, catchAsync(async (req, res) => {

    const { id } = req.params;

    const campground = *await* (Campground.findById(id))

    res.render('campgrounds/edit', { campground });

}))

router.put('/:id', validateCampground, reqLogin, catchAsync(async (req, res) => {

    const { id } = req.params;

    console.log(id);

    const cmpgrd = *await* Campground.findByIdAndUpdate(id, req.body.campground, { runValidators: true })

    res.redirect(`/campgrounds/${cmpgrd.\_id}`);

}))

router.get('/:id', catchAsync(async (req, res, next) => {

    const { id } = req.params;

    const campground = *await* Campground.findById(id).populate('reviews');

*if* (!campground) {

        req.flash('error', 'Campground not found');

        res.redirect('/campgrounds');

    }

    res.render('campgrounds/show', { campground });

}))

router.post('/', validateCampground, reqLogin, catchAsync(async (req, res, next) => {

    const { campground } = req.body;

    console.log(campground);

    const campgrounds = new Campground(campground);

*await* campgrounds.save();

    req.flash('success', 'Successfully created new Campground');

    res.redirect('/campgrounds');

}))

router.delete('/:id', reqLogin, catchAsync(async (req, res) => {

    const { id } = req.params;

*await* Campground.findByIdAndDelete(id);

    req.flash('success', 'Successfully delete Campground');

    res.redirect('/campgrounds');

}))

module.exports = router;

routes/review.js

const express = require('express');

const router = express.Router({ mergeParams: true });

const Campground = require('../models/campground');

const Review = require('../models/review')

const catchAsync = require('../utils/catchAsync');

const ExpressError = require('../utils/ExpressError');

const { reviewSchema } = require('../schema.js');

const { reqLogin } = require('../middleware');

const validateReview = (req, res, next) => {

    const { error } = reviewSchema.validate(req.body);

*if* (error) {

        const msg = error.details.map(el => el.message).join(',')

*throw* new ExpressError(400, msg)

    }

*else* {

        next();

    }

}

router.delete('/:rid', reqLogin, catchAsync(async (req, res) => {

    const { id, rid } = req.params;

*await* Review.findByIdAndDelete(rid);

*await* Campground.findByIdAndUpdate(id, { $pull: { reviews: { $in: [rid] } } })

    req.flash('success', 'Successfully delete a review');

    res.redirect(`/campgrounds/${id}`)

}))

router.post('/', validateReview, reqLogin, catchAsync(async (req, res, next) => {

    const { id } = req.params;

    const campground = *await* Campground.findById(id);

    const review = *await* new Review(req.body.review);

    campground.reviews.push(review);

*await* campground.save();

*await* review.save();

    req.flash('success', 'Successfully created a review');

    res.redirect(`/campgrounds/${id}`)

}))

module.exports = router;

views/partials/navbar.js

<nav *class*="navbar navbar-expand-lg navbar-dark bg-dark sticky-top">

    <div *class*="container-fluid">

        <a *class*="navbar-brand" *href*="#">Yelp-Camp</a>

        <button *class*="navbar-toggler" *type*="button" *data-bs-toggle*="collapse" *data-bs-target*="#navbarNavAltMarkup"

*aria-controls*="navbarNavAltMarkup" *aria-expanded*="false" *aria-label*="Toggle navigation">

            <span *class*="navbar-toggler-icon"></span>

        </button>

        <div *class*="collapse navbar-collapse" *id*="navbarNavAltMarkup">

            <div *class*="navbar-nav">

                <a *class*="nav-link" *href*="/">Home</a>

                <a *class*="nav-link" *href*="/campgrounds">All Campgrounds</a>

                <a *class*="nav-link" *href*="/campgrounds/new">New Campground</a>

            </div>

        </div>

        <div *class*="navbar-nav ml-auto">

            <% *if* (!currentUser) { %>

                <a *href*="/register" *class*="nav-link">Register</a>

                <a *href*="/login" *class*="nav-link">Login</a>

                <% } *else* { %>

                    <a *href*="/logout" *class*="nav-link">Log Out</a>

                    <% } %>

        </div>

    </div>

</nav>